URP Screen Space Reflections Guide

This document serves as an introduction and guide for the upcoming Screen Space Reflections feature for URP. The feature is currently in a usable preview state, ready for curious users to try out. We are looking to gather feedback before the initial release, and would love to hear what you think.

When you are ready to provide feedback, please take a moment to fill out this survey. For any additional questions or bug reports, please make a post on the discussions thread, or send an email to steven.kent@unity3d.com.

Thanks for your interest in trying out the feature, and for giving feedback.

What are Screen Space Reflections?

The Screen Space Reflection feature produces real-time glossy reflections by tracing rays in screen space and sampling the camera's color and depth buffers at various points along the ray. The technique can produce fairly high quality reflections of visible objects, which instantly update to reflect changes in the scene, without much setup or authoring steps required.

Screen Space Reflections can only use information that is visible on screen. When a reflection ray travels off screen, or behind an object, there is insufficient information to calculate a hit point. In those cases, a fallback cubemap will be used - either from a nearby reflection probe or from the skybox. This is imperfect, and may lead to artifacts. Screen Space Reflections are in general not a silver-bullet solution to specular lighting, but provide a decent approximation of it.

Enabling Screen Space Reflections

Setup Project

1. Install the latest Alpha version through the Unity Alpha program, at least 6000.6.0a7.

2. Create or load a URP project.

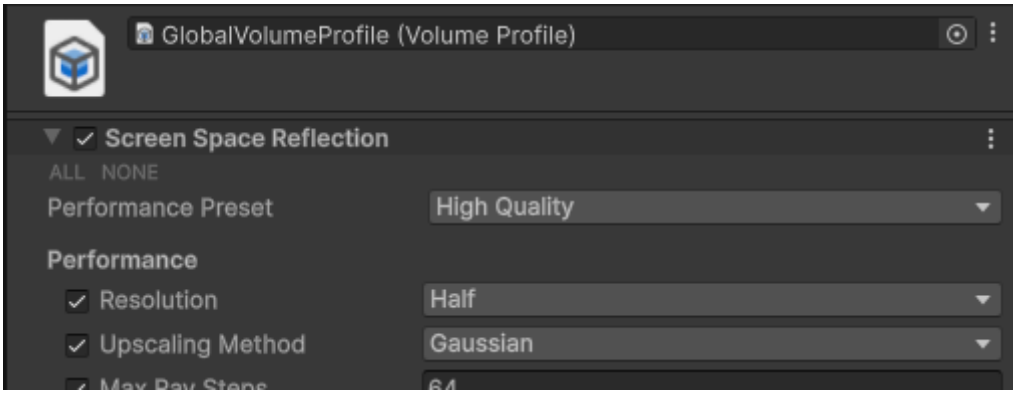
Configure Project

1. Navigate to Project Settings > Player > Other Settings.
2. Under Scripting Define Symbols, add URP_SCREEN_SPACE_REFLECTION as a new entry. This will make the Screen Space Reflection feature appear in URP's list of Renderer Features.

Add Renderer Feature

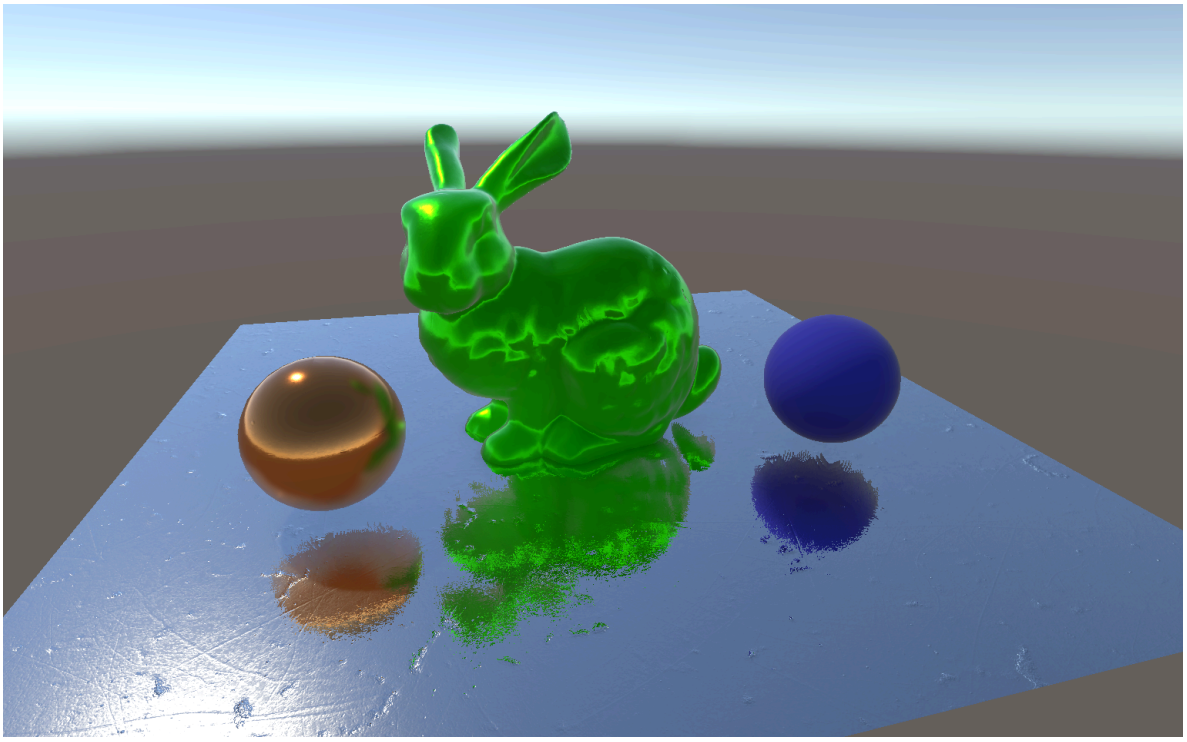
1. In your Universal Rendering Data asset, add the Screen Space Reflection Renderer Feature. This enables the feature for the given URP Renderer.

2. In a Volume Component, add a Screen Space Reflection Volume override. This is where you configure the behaviour and quality of Screen Space Reflections.



That's it!

You should now see realtime reflections on smooth surfaces.



Configuration

In this section, we describe the settings exposed by the Screen Space Reflection feature. Most of the relevant settings exist in the volume override.

Performance Settings

The settings in the **Performance** section of the volume override are the main ones to consider when tuning the performance of Screen Space Reflections. These settings can be set quickly using the **Performance Preset** dropdown, which contains some built-in presets. The settings are as follows:

- **Resolution:** The resolution to render reflections at. Higher resolution results in higher quality reflections, at the cost of performance. Screen Space Reflections can be quite expensive, especially when rendering at full resolution. Resolution is the most impactful setting for performance.
- **Upscaling Method:** When rendering at half or quarter resolution, this setting controls the method used for upscaling the reflections to full resolution.
    - Bilinear: Use hardware bilinear filtering. This is the cheapest method, but it may cause edges to look too 'soft', as reflections may be filtered across them.
    - Bilateral: Use bilinear filtering with bilateral weighting. This method is more expensive, but prevents filtering reflections across edges, which makes them look sharper.
- **Max Ray Steps:** The maximum amount of steps to take while tracing a reflection ray. Higher values result in higher quality reflections, at the cost of performance. When this setting is set too low, distant objects may not appear in reflections, and reflections may be 'cut off' or contain gaps.
- **Object Thickness:** How 'thick' each object should be when tracing reflection rays. The lower the value, the closer the ray must get to the object to be considered a hit. This setting mainly

affects appearance rather than performance. When set too low, reflections may appear noisy or contain gaps. When set too high, reflections may appear too thick, resulting in streaking artifacts near their edges. The setting should ideally be tuned to each scene.

- **Marching Method:** Which algorithm to use for tracing reflection rays. The performance characteristics of this setting depends largely on Max Ray Steps.

  Linear marching can produce reasonable looking reflections at much lower Max Ray Steps than Hierarchical marching. Hierarchical marching is much more efficient at higher Max Ray Steps, though, since rays can skip large portions of the screen at a time.

  Hierarchical depth marching requires building a hierarchical depth buffer, which adds an additional fixed performance penalty. We recommend using linear marching on lower-end devices like mobile phones, and hierarchical depth marching on higher-end devices like desktop and consoles.

  Hierarchical depth marching has the major benefit that it produces accurate reflections as long as the Max Ray Steps is set sufficiently high. Linear marching is more finicky, and requires tweaking some additional settings to get accurate reflections:

  - **Max Ray Length:** The maximum distance in world space units that a ray can travel. Linear marching divides the Max Ray Length into equally sized segments when tracing reflection rays. When the Max Ray Length is set too high, the segments become too large, causing rays to miss some valid hits. This may lead to missing reflections or gaps in reflections. This setting should be tuned to the scene, and set to value just large enough that reflections contain all the desired objects.
  - **Hit Refinement Steps:** Once a ray hits an object, the hit position is refined using binary search, making it more precise. This setting controls how many iterations of binary search to perform. Higher values decrease performance, but may produce more accurate looking reflections. Values between 1 and 5 are typically sufficient, with diminishing returns for larger values.
  - **Final Thickness Multiplier:** This setting is multiplied onto the Object Thickness to calculate a new thickness, which is used after the hit position has been refined. Lower values may produce more accurate looking reflections, but values that are too low can introduce gaps in reflections. This setting should be set to a value just large enough to eliminate gaps in reflections.

Visual Quality Settings

The settings in the **Visual Quality** section of the volume override are mostly artistic controls which affect the appearance of reflections. Some of these settings still have an impact on performance, but are not affected by the **Performance Preset** dropdown. The settings are as follows:

- **Mode:** Controls which objects should appear in reflections:
  - Disabled: No reflections are rendered, the feature is disabled.
  - Opaques Only: Only opaque objects appear in reflections.

    - Opaques And Transparents: Both opaque and transparent objects appear in reflections. This value incurs a performance penalty over only rendering opaque objects.

- **Reflection Strength:** Controls how strongly reflections contribute to the color at each pixel. A value of 0 causes reflections to disappear entirely, while a value 1 makes the reflections fully opaque.

- **Clamp Reflected Color:** When enabled, this setting clamp the color of reflections to be under a certain value. This is useful for mitigating reflections of very bright lighting, for example due to specular highlights.

    - **Max Color Value:** The maximum color value that a reflection is allowed to have in any color channel. Color values above the maximum are clamped.

- **Roughness Filter:** Controls the technique used to render rough/glossy reflections:

    - Disabled: No rough reflections appear. All reflections are mirror-like. This is the cheapest option.

    - Box Blur: A small box blur is used to blur reflections. This is faster, but produces worse looking rough reflections compared to Gaussian Blur.

    - Gaussian Blur: A gaussian blur is used to blur reflections. This is the most expensive option, but produces the best looking rough reflections.

- **Minimum Smoothness:** The minimum smoothness a material must have for it to receive Screen Space Reflections. Objects using materials with a smoothness value below this threshold only receive cubemap reflections from reflection probes or the skybox.

- **Smoothness Fade Start:** Reflections on objects using materials with smoothness values near to the Minimum Smoothness are gradually faded out to reduce popping. Reflections on objects with smoothness above the Smoothness Fade Start receive full strength reflections, while reflections on objects with smoothness in the range [Minimum Smoothness; Smoothness Fade Start] are gradually faded. Setting Minimum Smoothness and Smoothness Fade Start disables the fade.

- **Screen Edge Fade Distance:** Contribution from reflection rays with hit points near the edge of the screen are gradually faded, to avoid hard seams in reflections when some rays go off screen. This setting controls the size of the screen region where the fade occurs. Setting the value to 0 disables the fade entirely.

- **Normal Fade:** Contribution from reflection rays pointing towards the camera are gradually faded out to avoid artifacts. This setting controls how aligned the reflection ray direction and viewing direction must be for the fade to be applied. Setting the value to 0 disables the fade entirely.

- **Ray Length Fade:** Contribution from reflection rays with lengths that are near the Max Ray Length are gradually faded, to avoid hard seams in reflections when some rays go past the Max Ray Length. This setting controls the range of the fade in world space units. Higher values cause the fade to occur earlier. The maximum value for this setting is the current Max Ray Length, at which every ray is faded. This setting only has an effect when Linear Marching is enabled.

- **Reflect Sky:** Whether to use Screen Space Reflections to reflect the skybox where possible, or rely entirely on cubemap reflections from reflection probes or the skybox cubemap. Enabling this may produce higher quality reflections of the sky, but can introduce artifacts in some cases. We don't recommend enabling this setting when using Linear Marching, as the sky reflections may be distorted when the Max Ray Length is set too low.

- **Temporal Filtering:** Whether to use a temporal filter to stabilize reflections to reduce flickering and temporal instability. This is particularly useful when Temporal Anti-aliasing (TAA) is enabled, as this will result in the depth buffer being jittered, which can cause flickering. It's also useful when using Half or Quarter Resolution, as the aliasing introduced by upscaling can cause flickering when the camera moves.

    - **Base Blend Factor:** How strongly to weigh history contribution during temporal filtering. Higher values produce more stable reflections with less flickering, but may introduce ghosting artifacts.

After Opaque Toggle

The Screen Space Reflection Renderer Feature UI exposes a single setting, called **After Opaque**, which has implications for both performance and quality:

- **After Opaque disabled:** When the toggle is disabled (the default value), the Screen Space Reflection pass renders before the opaque render pass. This allows the contribution to be seamlessly integrated into the shading model in all rendering paths, and results in the higher quality reflections. However, it does impose a slight performance penalty, because the reflection color must be reprojected from the previous frame using motion vectors.

- **After Opaque enabled:** When the toggle is enabled, the Screen Space Reflection pass renders after the opaque pass. This is generally more performant, as the reflection color is taken from the current frame, rather than reprojected from the last one. However, it produces lower quality reflections, as they are composited onto the final frame after the shading model has been evaluated. In particular, the metallic value of each material is not taken into account, and certain visual effects like fresnel are diminished. When using this mode, the Reflection Strength volume setting can be particularly useful to dim reflections. We only recommend this setting for low-end devices.

Material Settings

Two additional settings are shown on the inspector of materials using shaders compatible with Screen Space Reflections. They are as follows:

- **Screen Space Reflection:** This toggle controls whether the material should receive Screen Space Reflections. When disabled, the no reflections are rendered on the objects using the material. Use this to disable reflections for objects where they are not suitable, or are of insufficient quality.

- **Contribute Screen Space Reflections:** This toggle controls whether the material should contribute to Screen Space Reflections, i.e. whether the material should be visible in reflections on other objects. This toggle only appears when the material is transparent; contribution is forced enabled for opaque objects. Use this to prevent artifacts from reflections of certain transparent effects, such as particle or VFX effects.

Troubleshooting

Below are some common pitfalls you may encounter while using Screen Space Reflections, and advice on how to solve them. Check the Known Issues section if your issue isn't listed.

- *Reflections are missing:* First make sure the feature is enabled by following the steps in the "Enabling Screen Space Reflections" section. Next, follow this checklist:
    - Check that the 'Mode' setting isn't 'Disabled'. If the object missing from reflections is transparent, ensure that the 'Mode' setting is set to 'Opaques And Transparents'.
    - Check that the 'Reflection Strength' setting is greater than 0.
    - Ensure that your reflective material has a smoothness value higher than the 'Minimum Smoothness' setting.
    - If 'Linear Marching' is enabled, ensure that the 'Final Thickness Multiplier' is greater than 0.
    - Ensure that the 'Screen Space Reflection' toggle is enabled on the reflective material.
    - Try increasing the values of the 'Max Ray Steps' and 'Max Ray Length' settings.
    - Try increasing the value of the 'Object Thickness' setting.
    - Try setting all the fade-related settings to their minimum possible values.

- *Reflections contain gaps or holes:* The 'Max Ray Steps' or 'Object Thickness' settings may be set too low, so try increasing them. If you have 'Linear Marching' enabled, the 'Max Ray Length' and 'Final Thickness Multiplier' must also be set sufficiently high, and may need to be increased.

- *Reflections appear to be abruptly cut off:* This is typically caused by the 'Max Ray Steps' setting being too low, or by rays that travel off screen or behind objects, where there is insufficient information to calculate a hit. First try increasing the step count. If that doesn't help, the best option is to try to hide the seam using the various fade-related settings.

- *Reflections appear stretched or have streaky artifacts:* The 'Object Thickness' setting may be set too high. This setting should be tuned to the scene, and set to a value *just* large enough that reflections appear coherent, without gaps or holes. If you are using 'Linear Marching', this can also sometimes be caused by the 'Max Ray Length' or 'Hit Refinement Steps' being set too low, so try increasing those.

- *Reflections are vibrating and flickering:* The typical culprits are jittering caused by Temporal Anti-aliasing (TAA) or aliasing caused by rendering at Half or Quarter Resolution. To mitigate this, try enabling the 'Temporal Filtering' setting and tweaking the 'Base Blend Factor'.

- *Performance is bad:* Though we are still working on improving performance, Screen Space Reflections are inherently quite expensive. The best ways to improve performance involve doing less work, usually at the cost of quality. The most impactful settings to tweak are 'Resolution', 'Max Ray Steps', 'Linear Marching', 'Upscaling Method' and 'Rough Reflections'.

Limitations

Below is a list of current limitations, some of which may be addressed in future releases.

- Rough reflections are implemented by blurring the reflected color texture in screen space. This produces less physically plausible reflections than more modern techniques based

shooting rays in random directions and temporally accumulating the result. In the future, we may add a more physically correct accumulation-based mode for higher-end platforms.

- Reflection rays that travel off screen do not contribute, and a fallback cubemap is used instead (either from a nearby reflection probe, or from the sky). We do not support continuing off-screen rays with traditional raytracing for more accurate reflections in this release.
- Screen Space Reflections are not supported for the WebGL build target. To target browsers, consider using WebGPU, which supports the feature.
- Hierarchical depth marching requires Compute Shader support. On platforms that don't support Compute Shaders, the Linear Marching setting is automatically forced enabled.
- Semi-transparent objects with Screen Space Reflections enabled are effectively treated as opaque when tracing reflection rays. If a ray hits a part of the object with low or 0 alpha, the hit still counts. This can cause artifacts for some transparent objects like particles or VFX, which can be mitigated with the per-material Screen Space Reflection toggles. Alpha cut-outs, like foliage, are fully supported - rays won't hit the parts that have been cut out.

Known Issues

Below is a list of known issues and bugs. If you encounter any issues or bugs not listed here, please let us know!

- Reflections may appear slightly delayed or laggy in the Scene View, because URP can't reliably produce motion vectors for the Scene View camera. For the same reason, Temporal Anti-aliasing (TAA) doesn't work in the Scene View. We are working on fixing this issue.
- Dynamic Resolution is not yet supported. Setting a constant Render Scale should work fine, but varying it dynamically can cause reflections to flicker. We are working on fixing this issue.
- Orthographic cameras are not supported. They do not render any reflections.

Custom Shader Support

The following built-in URP shaders are fully compatible with Screen Space Reflections:

- Lit
- Complex Lit
- Particles Lit
- Terrain Lit
- SpeedTree shaders
- Lit ShaderGraph shaders

All shaders that have a DepthNormals pass can *contribute* to Screen Space Reflections. This covers the majority of shaders made for URP, both built-in and third-party.

To *receive* Screen Space Reflections, custom shaders must be updated to support the feature:

First, the UniversalForward and UniversalGBuffer pass must declare the _SCREEN_SPACE_REFLECTION keyword, like so:

```C/C++
Pass
{
    Name "ForwardLit"
    Tags { "LightMode" = "UniversalForward" }
    HLSLPROGRAM
    ...
    #pragma multi_compile_fragment _ _SCREEN_SPACE_REFLECTION
    ...
    ENDHLSL
}

Pass
{
    Name "GBuffer"
    Tags{ "LightMode" = "UniversalGBuffer" }
    HLSLPROGRAM
    ...
    #pragma multi_compile_fragment _ _SCREEN_SPACE_REFLECTION
    ...
    ENDHLSL
}
```

Next, the DepthNormals must declare the _WRITE_SMOOTHNESS keyword:

```C/C++
Pass
{
    Name "DepthNormals"
    Tags { "LightMode" = "DepthNormals" }
    HLSLPROGRAM
    ...
    #pragma multi_compile_fragment _ _WRITE_SMOOTHNESS
    ...
    ENDHLSL
}
```

Finally, the fragment function for the DepthNormals must output smoothness to the alpha channel when _WRITE_SMOOTHNESS is enabled:

```C/C++
void DepthNormalsFragment(Varyings input, out half4 outNormalWS : SV_Target0)
{
    ...
    #if defined(_WRITE_SMOOTHNESS)
    float alpha = SampleAlbedoAlpha(input.uv, TEXTURE2D_ARGS(_BaseMap, sampler_BaseMap)).a;
    outNormalWS.a = SampleMetallicSpecGloss(input.uv, alpha).a;
    #endif
}
```

A working implementation of this can be found in com.unity.render-pipelines.universal\Shaders\LitDepthNormalsPass.hlsl, which custom shaders may choose to include.

With those changes, any shader that samples environment reflections (using the GlossyEnvironmentReflection function) should receive Screen Space Reflections. You can also sample the reflections manually using the GetScreenSpaceReflection function in com.unity.render-pipelines.universal/ShaderLibrary/SampleScreenSpaceReflection.hlsl.

FAQ

Does the Screen Space Reflections feature work together with Surface Cache GI?
Yes, the features can be mixed freely. Surface Cache provides diffuse lighting, while Screen Space Reflections provide specular lighting. The features simply add their contributions together.

Can Screen Space reflections be used with baked lighting and/or reflection probes?
Screen Space Reflections are applied additively on top of existing baked lighting, and are fully compatible with it. In cases when rays travel off screen or behind objects, reflections are instead calculated using nearby reflection probes, if there are any.

Does the Screen Space Reflections feature require hardware ray-tracing?
No, we do not require hardware raytracing, nor do we use any kind of software equivalent. The technique works entirely by marching rays in screen space. In the future, we may extend the feature with raytracing capabilities for greater fidelity.